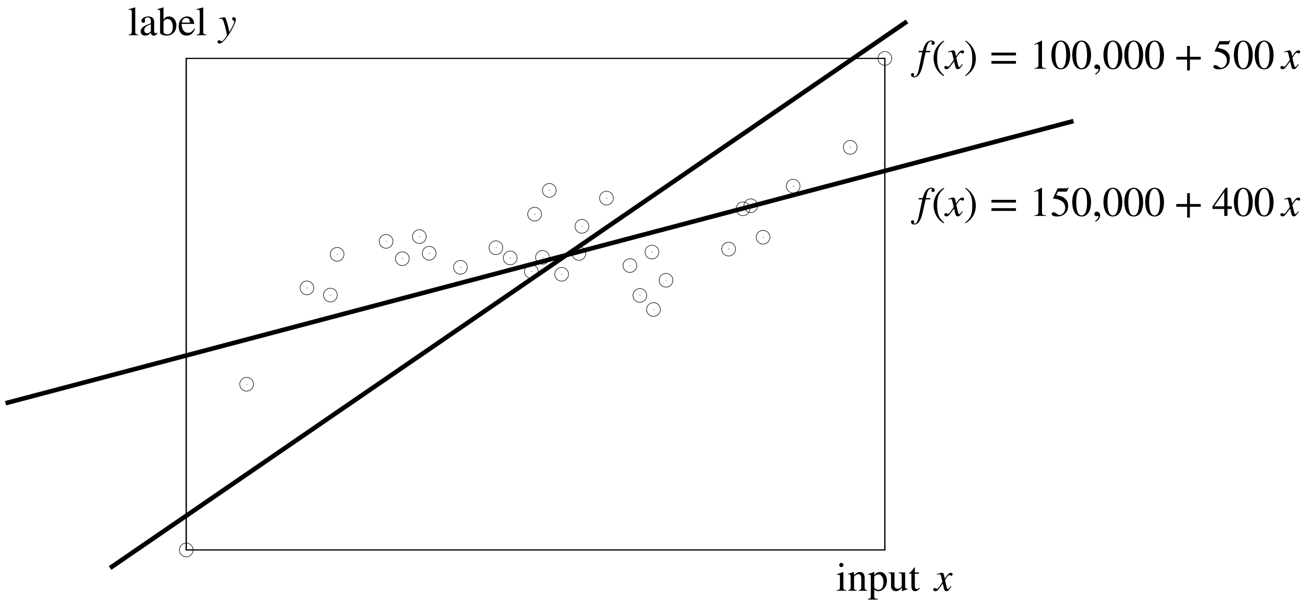
- HW0 due Tuesday midnight
- Extra office hours
  - **Monday:**
    - Tim Li, 10:30 - 11:30
    - **Sewoong Oh, 12:30 - 1:30**
    - Hugh Sun, 14:30 - 15:30
  - **Tuesday:**
    - Josh Gardner, 9:00 - 10:00
    - Hugh Sun, 14:30 - 15:30
    - Jakub Filipek, 16:00 - 17:00
    - **Pemi Nguyen, 17:00 - 20:00**

Lecture 4: Polynomial regression

- How to fit more complex data?

Recap: Linear Regression



- In general high-dimensions, we fit a linear model with intercept

  $y_i \simeq w^T x_i + b$, or equivalently $y_i = w^T x_i + b + \epsilon_i$

  with model parameters $(w \in \mathbb{R}^d, b \in \mathbb{R})$ that minimizes ℓ_2-loss

$$\arg\min_{w,b} \quad \mathcal{L}(w,b) = \sum_{i=1}^{n} \underbrace{(y_i - (w^T x_i + b))^2}_{\text{error } \epsilon_i}$$

Recap: Linear Regression

- The least squares solution, i.e. the minimizer of the ℓ_2-loss can be written in a **closed form** as a function of data $\mathbf{X}$ and $\mathbf{y}$ as

$$\mathcal{L}(w,b) = \sum_{i=1}^{n} \left(y_i - w^T x_i - b \right)^2$$

As we derived in class:

$$\mu = \frac{1}{n}\mathbf{X}^T\mathbf{1}$$

$$\widetilde{\mathbf{X}} = \mathbf{X} - \mathbf{1}\mu^T$$

$$\widehat{w}_{\text{LS}} = (\widetilde{\mathbf{X}}^T\widetilde{\mathbf{X}})^{-1}\widetilde{\mathbf{X}}^T\mathbf{y}$$

$$\widehat{b}_{\text{LS}} = \frac{1}{n}\sum_{i=1}^{n} y_i - \mu^T\widehat{w}_{\text{LS}} \quad \in \mathbb{R}$$

or equivalently using straightforward linear algebra by setting the gradient to zero:

$$\begin{array}{c} d \\ 1 \end{array}\begin{bmatrix} \widehat{w}_{\text{LS}} \\ \widehat{b}_{\text{LS}} \end{bmatrix} = \left(\begin{bmatrix} \mathbf{X}^T \\ \mathbf{1}^T \end{bmatrix} [\mathbf{X} \quad \mathbf{1}] \right)^{-1} \begin{bmatrix} \mathbf{X}^T \\ \mathbf{1}^T \end{bmatrix} \mathbf{y}$$

linear Algebra

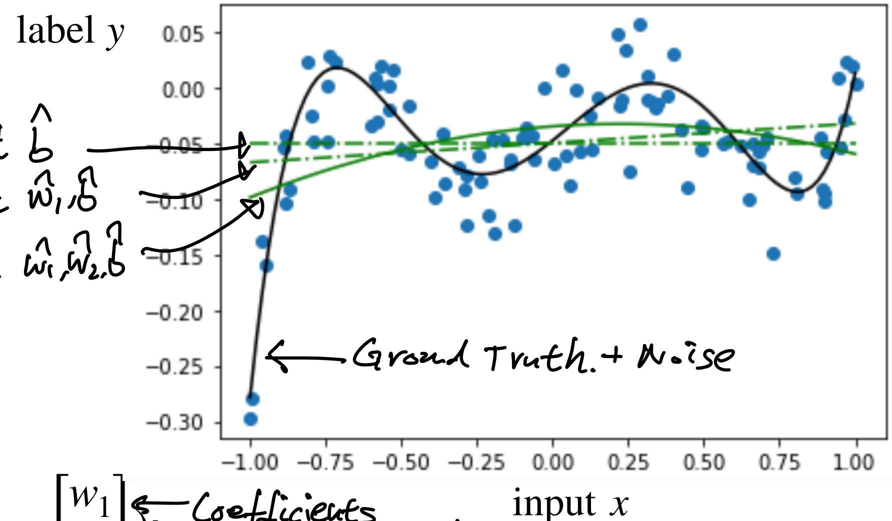
Quadratic regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

fit $\hat{b}$

fit $\hat{w}_1, \hat{b}$

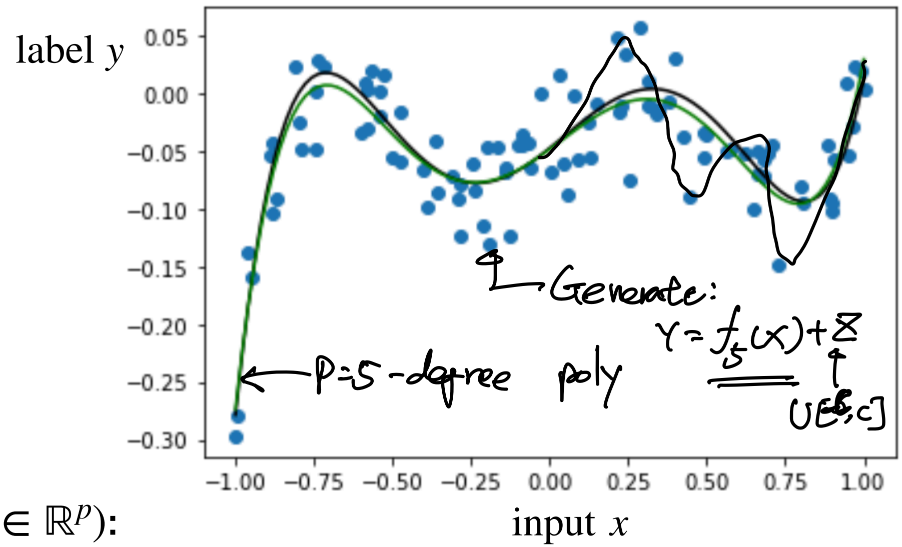
fit $\hat{w}_1, \hat{w}_2, \hat{b}$



- **Linear model with parameter** (b, w_1): $\;\underset{\mathbb{R}}{b}, \underset{\mathbb{R}}{w_1}$
  - $y_i = b + w_1 x_i + \epsilon_i$
  - $\mathbf{y} = \mathbf{1}b + \mathbf{X}w_1 + \epsilon$
- **Quadratic model with parameter** $(\underset{\mathbb{R}}{b}, w = \begin{bmatrix} w_1 \\ w_2 \end{bmatrix})$: Coefficients of polynomial
  - $y_i = b + w_1 x_i + w_2 x_i^2 + \epsilon_i$
  - Define $h : \mathbb{R} \to \mathbb{R}^2$ such that $x \mapsto h(x) = \begin{bmatrix} x \\ x^2 \end{bmatrix}$
  - $y_i = b + \underbrace{h(x_i)^T w}_{= w_1 x_i + w_2 x_i^2} + \epsilon_i$
  - Treat $h(x)$ as new input features. Let $\mathbf{H} = \begin{bmatrix} h(x_1)^T \\ \vdots \\ h(x_n)^T \end{bmatrix}$. Replace x_i by $\begin{bmatrix} x_i \\ x_i^2 \end{bmatrix}$
  - $\mathbf{y} = \mathbf{1}b + \mathbf{H}w + \epsilon$

Degree-p polynomial regression in 1-dimension

- **Data:** $\mathbf{X} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$



- **Linear model with parameter** (b, w_1)**:**
  - $y_i = b + w_1 x_i + \epsilon_i$
  - $\mathbf{y} = \mathbf{1}b + \mathbf{X}w_1 + \epsilon$
- **Degree-p model with parameter** $(b, w \in \mathbb{R}^p)$**:**
  - $y_i = b + w_1 x_i + \cdots + w_p x_i^p + \epsilon_i$
  - Define $h : \mathbb{R} \to \mathbb{R}^p$ such that $x \mapsto h(x) = \left.\begin{bmatrix} x \\ \vdots \\ x^p \end{bmatrix}\right\} \mathbb{R}^p$
  - $y_i = b + h(x_i)^T w + \epsilon_i$
  - Treat $h(x)$ as new input features and let $\mathbf{H} = \begin{bmatrix} h(x_1)^T \\ \vdots \\ h(x_n)^T \end{bmatrix}$
  - $\mathbf{y} = \mathbf{1}b + \mathbf{H}w + \epsilon$

hard coded

know

* larger p, we get complex models

Q. which p to use?

Q. what is down side for larger p?

Q. what is down side for smaller p?

Degree-p polynomial regression in d-dimension

let $d=2, p=3$

$$h(x) = \begin{bmatrix} x_1 \\ x_2 \\ x_1^2 \\ x_2^2 \\ x_1^3 \\ x_2^3 \\ x_1x_2 \\ x_1^2x_2 \\ x_1x_2^2 \end{bmatrix}$$

feature engineering

$\sin(ax+b)$

$e^{-(ax+b)^2}$

$\log(ax+b)$

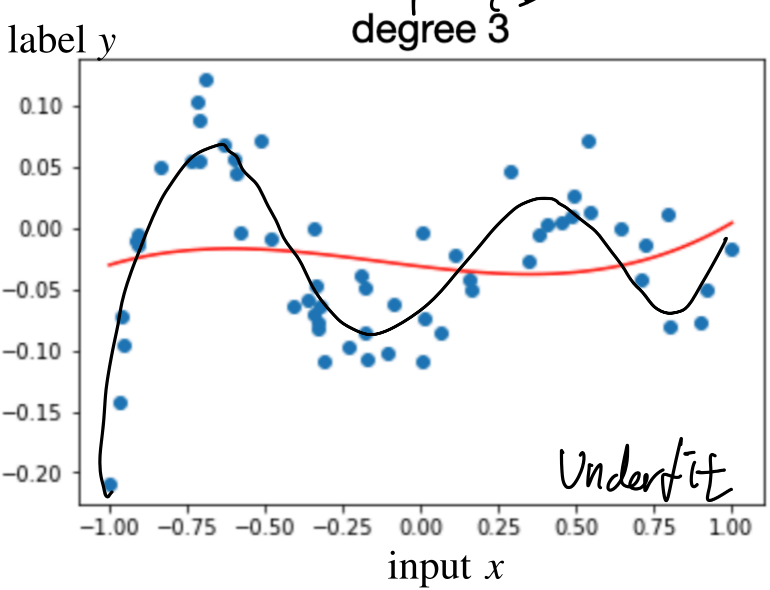
- **Data:** $\mathbf{X} = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1d} \\ & x_2^T & & \\ & \vdots & & \\ & x_n^T & & \end{bmatrix}$, $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$
- **Degree-p model with parameter $(b, w \in \mathbb{R}^{dp})$:**
  - $y_i = b + x_i^T \underset{\in \mathbb{R}^d}{w_1} + \cdots + (x_i^p)^T \underset{\in \mathbb{R}^d}{w_p} + \epsilon_i$, where $x_i^P = \begin{bmatrix} x_{i1}^p \\ \vdots \\ x_{id}^p \end{bmatrix}$
  - Define $h : \mathbb{R}^d \to \mathbb{R}^{dp}$ such that $x \mapsto h(x) = \begin{bmatrix} x \\ \vdots \\ x^p \end{bmatrix} \in \mathbb{R}^{dp}$
  - $y_i = b + h(x_i)^T w + \epsilon_i$
  - Treat $h(x)$ as new input features and let $\mathbf{H} = \left.\begin{bmatrix} h(x_1)^T \\ \vdots \\ h(x_n)^T \end{bmatrix}\right\} \mathbb{R}^{d \cdot p}$
  - $\mathbf{y} = \mathbf{1}b + \mathbf{H}w + \epsilon$
- In general, any feature $h(x)$ can be used, e.g., $\sin(ax+b)$, $e^{-b(x-a)^2}$, $\log x$, etc.

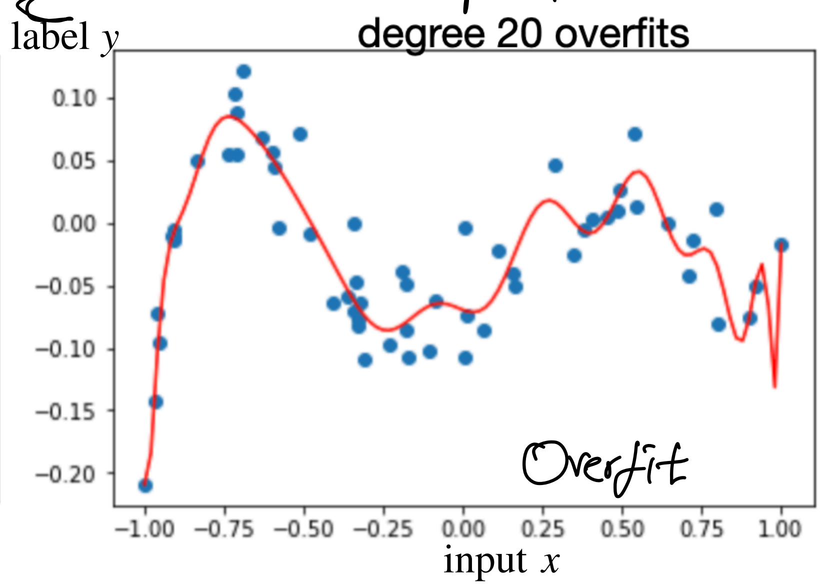
Which p should we choose?

- First instance of class of models with different representation power = $\boxed{\text{model complexity}}$

$w \in \mathbb{R}^3$ Complexity ↓

$w \in \mathbb{R}^{20}$ Complexity ↑

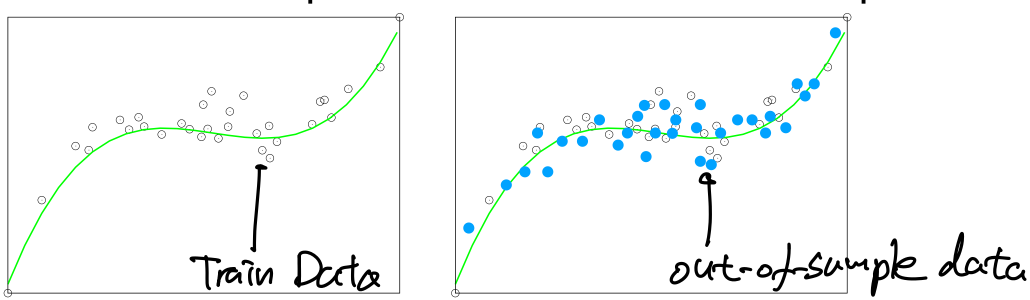




- How do we determine which is better model?

Generalization

- we say a predictor **generalizes** if it performs as well on unseen data as on training data
- formal mathematical definition involves probabilistic assumptions (coming later in this week)
- the data used to train a predictor is **training data** or **in-sample data**
- we want the predictor to work on **out-of-sample data**
- we say a predictor **fails to generalize** if it performs well on in-sample data but does not perform well on out-of-sample data



- **train** a cubic predictor on 32 (**in-sample**) white circles: Mean Squared Error (MSE) 174
- **predict** label y for 30 (**out-of-sample**) blue circles: MSE 192

$174 \simeq 192$

- conclude this predictor/model generalizes, as in-sample MSE $\simeq$ out-of-sample MSE

Split the data into training and testing

- a way to mimic how the predictor performs on unseen data
- given a single dataset $S = \{(x_i, y_i)\}_{i=1}^{n}$
- we split the dataset into two: training set and test set
- selection of data train/test should be done randomly (80/20 or 90/10 are common)
- **training set** used to train the model
  - minimize $\mathscr{L}_{\text{train}}(w) = \frac{1}{|S_{\text{train}}|} \sum_{i \in S_{\text{train}}} (y_i - x_i^T w)^2$ $\leftarrow MSE_{train}$
    $h(x_i)^T \cdot w + b$
- **test set** used to evaluate the model
  - $\mathscr{L}_{\text{test}}(w) = \frac{1}{|S_{\text{test}}|} \sum_{i \in S_{\text{test}}} (y_i - x_i^T w)^2$ $\leftarrow MSE_{test}$
- this assumes that test set is similar to unseen data ← random split
- **test set should never be used in training** + hyperparameter tuning

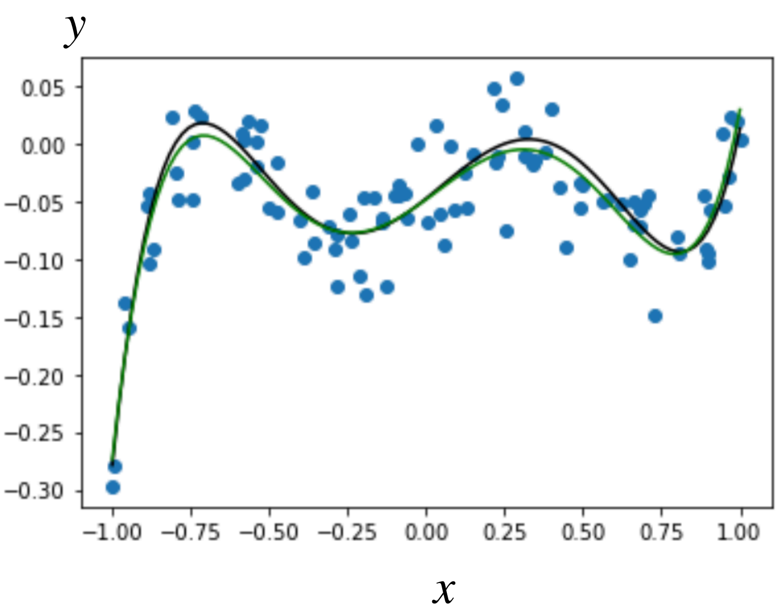
We say a model w or predictor **overfits** if $\mathscr{L}_{\text{train}}(w) \ll \mathscr{L}_{\text{test}}(w)$

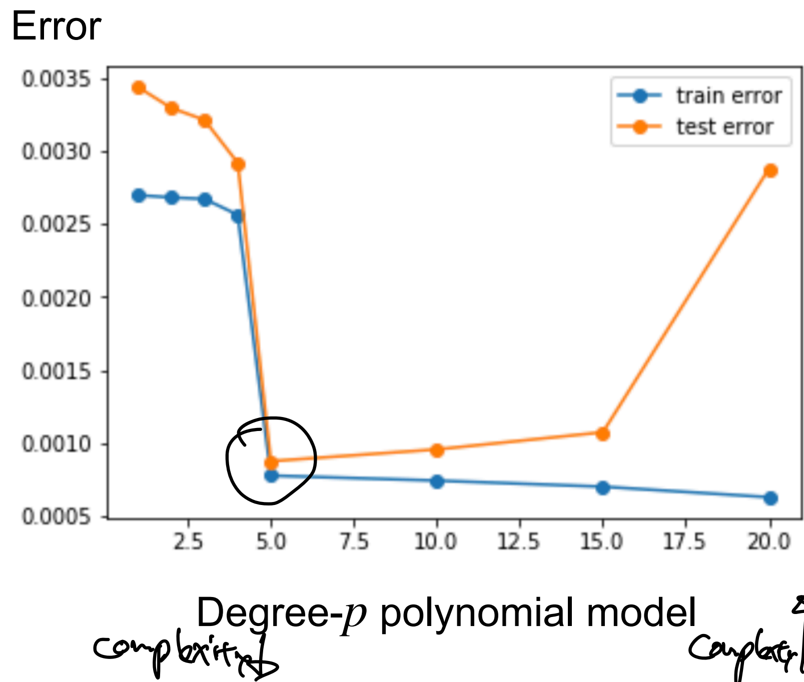
	small training error	**large training error**
small test error	**generalizes well** **performs well**	**possible, but unlikely**
large test error	**fails to generalize** **Overfitting**	**generalizes well** **performs poorly**

decrease complexity
regularization

increase complexity
e.g. adding more feature

How do we choose which model to use?



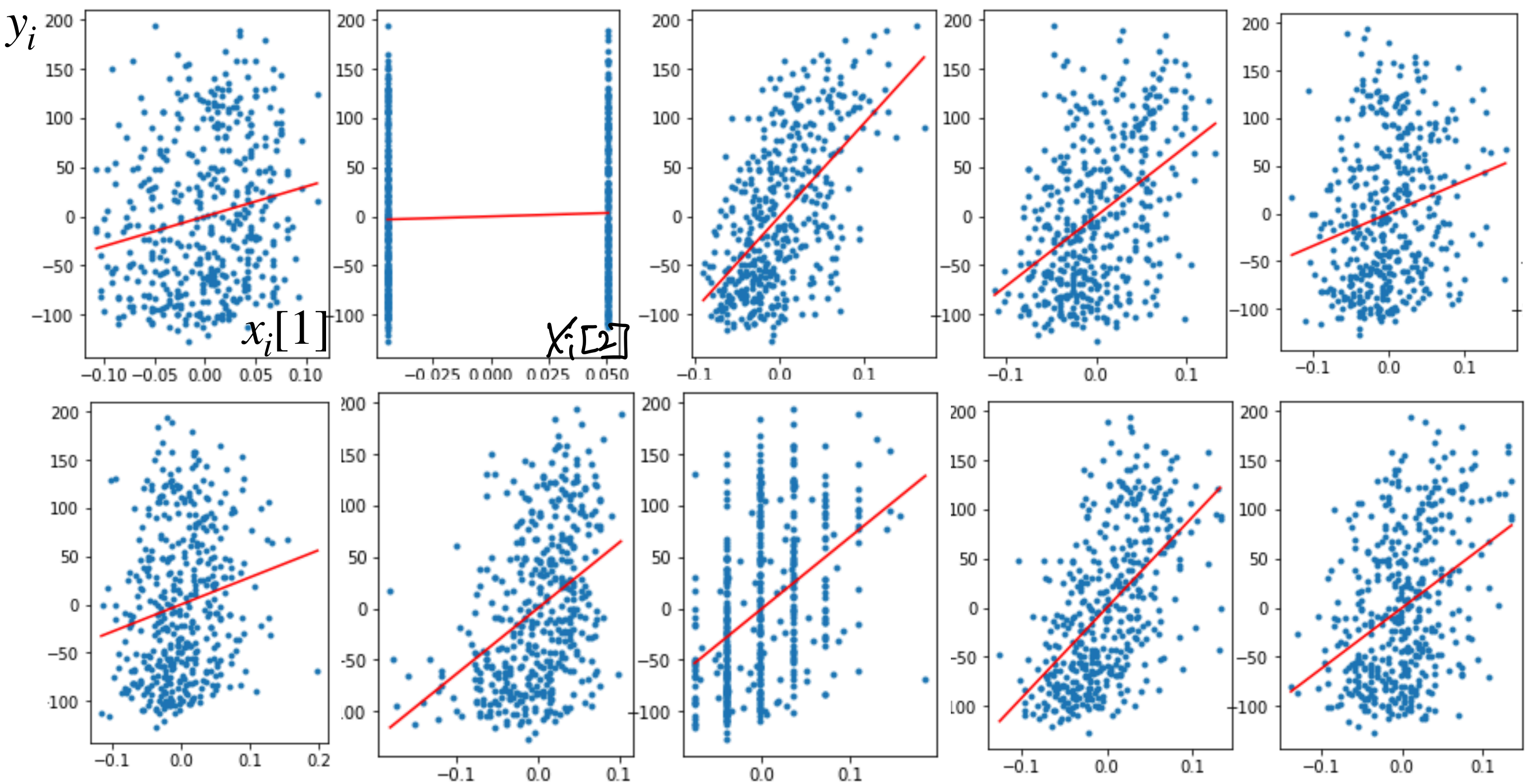


1. first use 60 data points to train and 60 data points to test and train several models to get the above graph on the right
2. then choose degree $p = 5$, since it achieves **minimum test error**
3. now re-train on all 120 data points with degree 5 polynomial model

demo2_lin.ipynb

Another example: Diabetes

- Example: Diabetes
  - 10 explanatory variables
  - from 442 patients
  - we use half for train and half for validation



Mean Squared Error

most complex

Features	Train MSE	Test MSE
All	2640	3224
S5 and BMI	3004	3453
S5	3869	4227
BMI	3540	4277
S4 and S3	4251	5302
S4	4278	5409
S3	4607	5419
None	5524	6352

← all vars.

← 2 vars

← best single variable

← fit b

Ohbay A.

- **test MSE is the primary criteria for model selection**
- Using only 2 features (S5 and BMI), one can get very close to the prediction performance of using all features
- Combining S3 and S4 does not give any performance gain

demo3_diabetes.ipynb

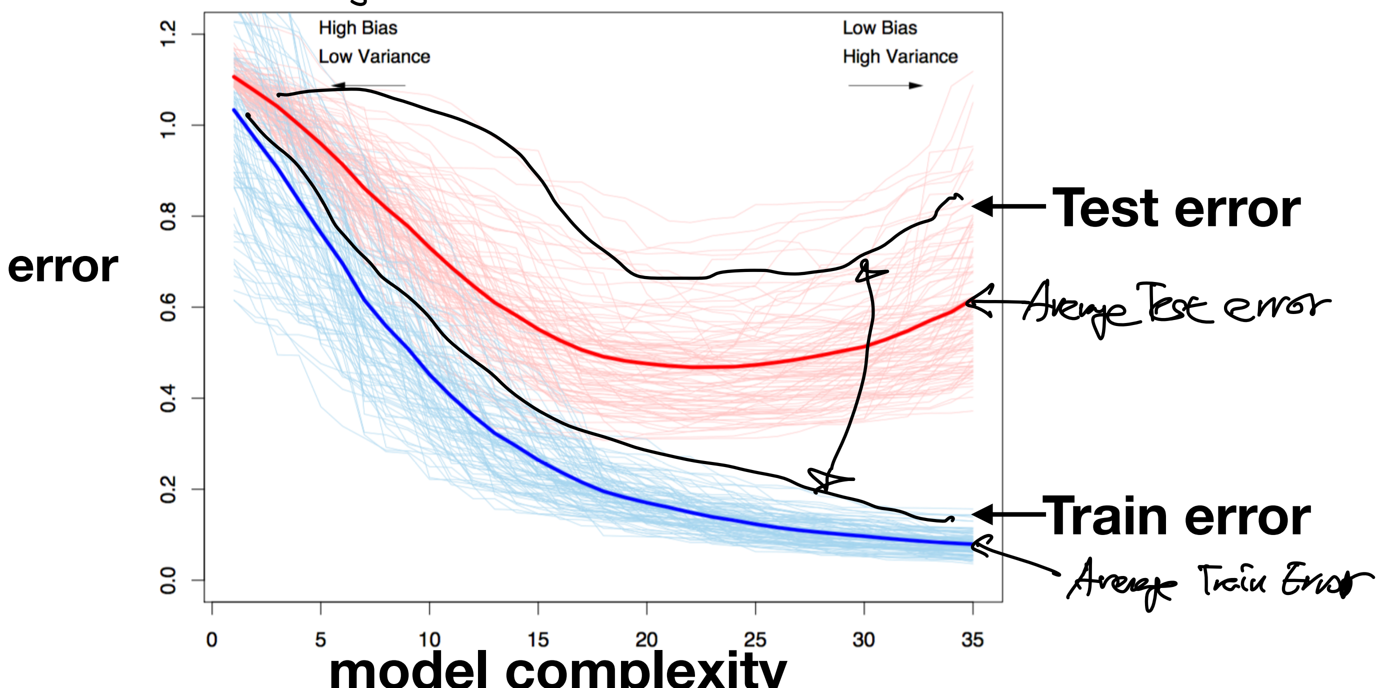
What does the bias-variance theory tell us?

$*$ Assume: $\exists\, P_{X,Y} \to (x_i, y_i)$ drawn iid

- **Train error** (random variable, randomness from $\mathscr{D}$)
  - Use $\mathscr{D} = \{(x_i, y_i)\}_{i=1}^{n} \sim P_{X,Y}$ to find $\widehat{w}$
  - Train error: $\mathscr{L}_{\text{train}}(\widehat{w}) = \dfrac{1}{|\mathscr{D}|} \sum_{(x_i,y_i)\in\mathscr{D}} (y_i - \widehat{w}^T x_i)^2 \in \mathbb{R}$
- ~~recall~~ the **test error** is an unbiased estimator of the **true error**
- **True error** (random variable, randomness from $\mathscr{D}$)
  - True error: $\mathscr{L}_{\text{true}}(\widehat{w}) = \underbrace{\mathbb{E}_{(x,y)\sim P_{X,Y}}[(y - \widehat{w}^T x)^2]}_{\text{R.V}}$ $\in \mathbb{R}$, Unknown because we do not have $P_{X,Y}$
- **Test error** (random variable, randomness from $\mathscr{D}$ and $\mathscr{T}$)
  - Use $\mathscr{T} = \{(x_i, y_i)\}_{i=1}^{m} \sim P_{X,Y}$ ← we access only samples from $P_{X,Y}$
  - Test error: $\mathscr{L}_{\text{test}}(\widehat{w}) = \dfrac{1}{|\mathscr{T}|} \sum_{(x_i,y_i)\in\mathscr{T}} (y_i - \widehat{w}^T x_i)^2$
- theory explains **true error**, and hence expected behavior of the (random) **test error**

What does the bias-variance theory tell us?

- Train error is optimistically biased (i.e. smaller) because the trained model is minimizing the train error
- Test error is unbiased estimate of the true error, if test data is never used in training a model or selecting the model complexity
- Each line is an i.i.d. instance of $\mathscr{D}$ and $\mathscr{T}$



Questions?

Questions?

Questions?